Simulationen und Mathematische Programmierung mit SAS

Gerhard Svolba
SAS Austria
Mariahilfer Straße 116
A-1070 Wien
Sastools.by.gerhard@gmx.net

Zusammenfassung

SAS ist bekannt für mächtiges Datenmanagement und umfassende Auswerte-Prozeduren und deckt mit seinem Portfolio den gesamten Analytik Life-Cycle ab. Mit SAS können aber genauso umfassende Simulations-Szenarien aufgebaut, durchgeführt und analysiert werden.
Dieser Beitrag zeigt die Mächtigkeit der SAS Software im Bereich der Monte-Carlo-Simulationen. Es wird gezeigt, welche Funktionen und Methoden in Base SAS und SAS IML für die Erzeugung von Zufallszahlen, die Approximation von Verteilungen und die Iteration über unterschiedliche Szenarien besonders wichtig sind.
Es werden SAS Code Beispiele und Tipps und Tricks gezeigt, wie Simulationen einfach und performant aufgesetzt und durchgeführt werden können. Dies beinhaltet auch Möglichkeiten der mathematischen Programmierung mit SAS, wo Sie u.a. Matrizenoperationen auf eigene Daten direkt in SAS anwenden können.

Schlüsselwörter: Simulationen, Monte Carlo Analysen, SAS IML, PROC IML, Zufallszahlen, RAND-Funktion, PROC SIMNORMAL

1 Einleitung

1.1 Verwendung von SAS für Simulations-Studien

Die SAS Software bietet vielfältige Möglichkeiten für die Durchführung von Simulationen und Simulationsstudien. Dieser Beitrag fokussiert sich auf die Möglichkeiten in folgenden 3 Bereichen:

- Die Verwendung des SAS Datastep für die Simulation von Daten aus univariaten und unkorrelierten multivariaten Verteilungen
- Die Nutzung der SAS IML Software für die Simulation von Daten aus vielen Verteilungen zum Beispiel aus korrelierten multivariaten Verteilungen, oder die Definition neuer Funktionen für das Erzeugen von Verteilungen, die in SAS nicht vorhanden sind
- Procedures in SAS STAT und SAS ETS zur Simulation von Daten mit speziellen Eigenschaften, wie zum Beispiel PROC SIMNORMAL für die Simulation von Daten aus multivariaten Normalverteilungen, PROC SIM2D für die räumliche Simulation von Daten in Gaussian Random Fields, oder die PROC COPULA für

multivariate Verteilungen, wo die Abhängigkeitsstruktur von der marginal Verteilung isoliert wird.

Darüber hinaus gibt es noch Simulationsmöglichkeiten in SAS, die in diesem Vortrag nicht behandelt werden.

- Das SAS Simulation Studio aus SAS OR für die Simulation von diskreten Ereignissen.
- Proc MCMC aus SAS STAT eine allgemeine Markov-Chain Monte Carlo Prozedur zum Schätzen Bayesianischer Modelle.
- Proc Risk und SAS Risk Management zur Simulation von Risiko Parametern.
- Proc MODEL aus SAS ETS für die Monte Carlo Simulation von Zeitreihenmodellen.

1.2 Simulationsbeispiel: Konsequenz schlechter Datenqualität auf die Modellgüte

In Buch „Data Quality for Analytics Using SAS“ [1] werden zahlreiche Simulationsstudien gezeigt, welche die Konsequenz schlechter Datenqualität auf die Modellgüte von prädiktiven Modellen sowie von Zeitreihenmodellen zeigt.

Neben wichtigen Erkenntnissen über den Effekt schlechter Datenqualität auf die Modellgüte zeigen diese Simulationsmodelle auch eindrucksvoll auf, wie unterschiedliche Komponenten des SAS Systems verwendet und kombiniert werden können, um Simulationsstudien optimal durchzuführen.

In diesen Simulationsstudien werden Datenmanagement Anweisungen in SAS Datasteps mit analytischen Prozeduren aus SAS STAT, SAS ETS oder SAS High Performance Forecasting kombiniert und die Simulationsergebnisse mit deskriptiven und graphischen Auswertemöglichkeiten in SAS zusammengefasst und dargestellt.

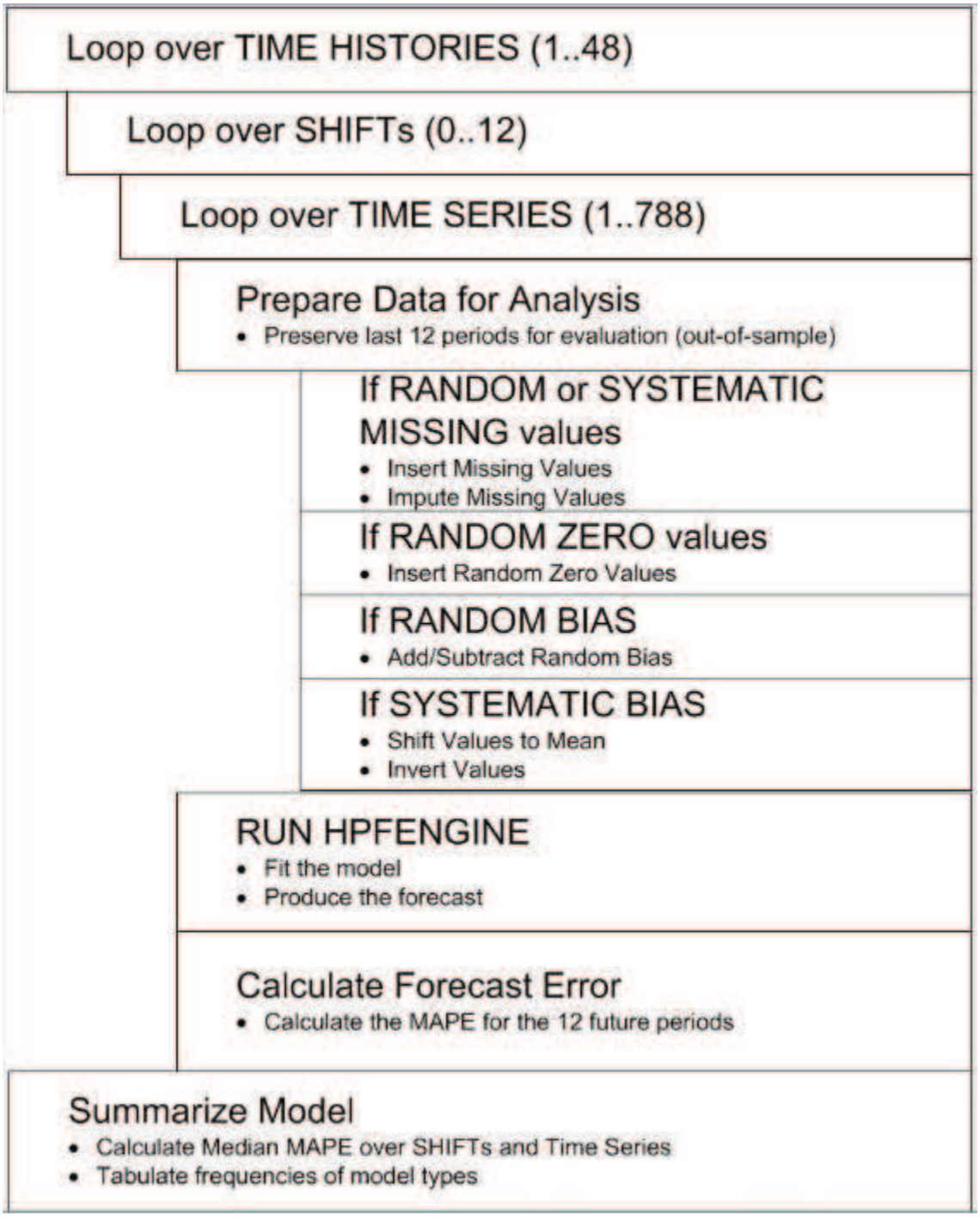

Abbildung 1: Ablauf für die Simulationen in Zeitreihenmodellen

Im Bereich des Predictive Modeling werden Simulationsmöglichkeiten im SAS Enterprise Miner gezeigt. Abbildung 2 illustriert beispielhaft einen solchen Ablauf. Der Vorteil der Kombination der Simulation mit dem SAS Enterprise Miner ist die Nutzbarkeit von Komponenten zur Modellentwicklung (Regressions-Knoten), der Imputation von fehlenden Werden (Impute Missing Values Knoten), sowie der Modellvalidierung (Model Comparison Knoten). Unterstützt wird diese Möglichkeiten im SAS Enterprise Miner durch den SAS Code Knoten, wo individuelle Simulationsanweisungen und Statements eingegeben werden können sowie dem Start- und End-Groups Knoten, der eine Iteration über die Simulationsszenarien ermöglicht.

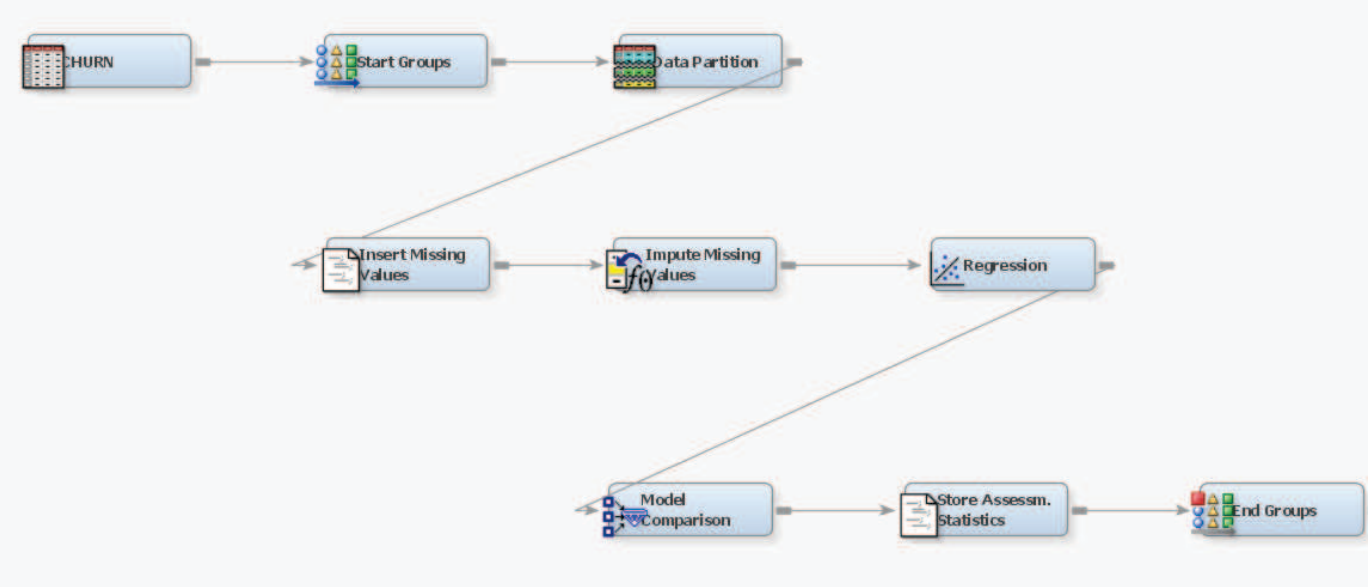

Abbildung 2: Simulationsablauf im SAS Enterprise Miner

Ergebnisse können hier wieder direkt über SAS Auswertemöglichkeiten, wie der PROC SGRENDER und der PROC TEMPLATE dargestellt werden.

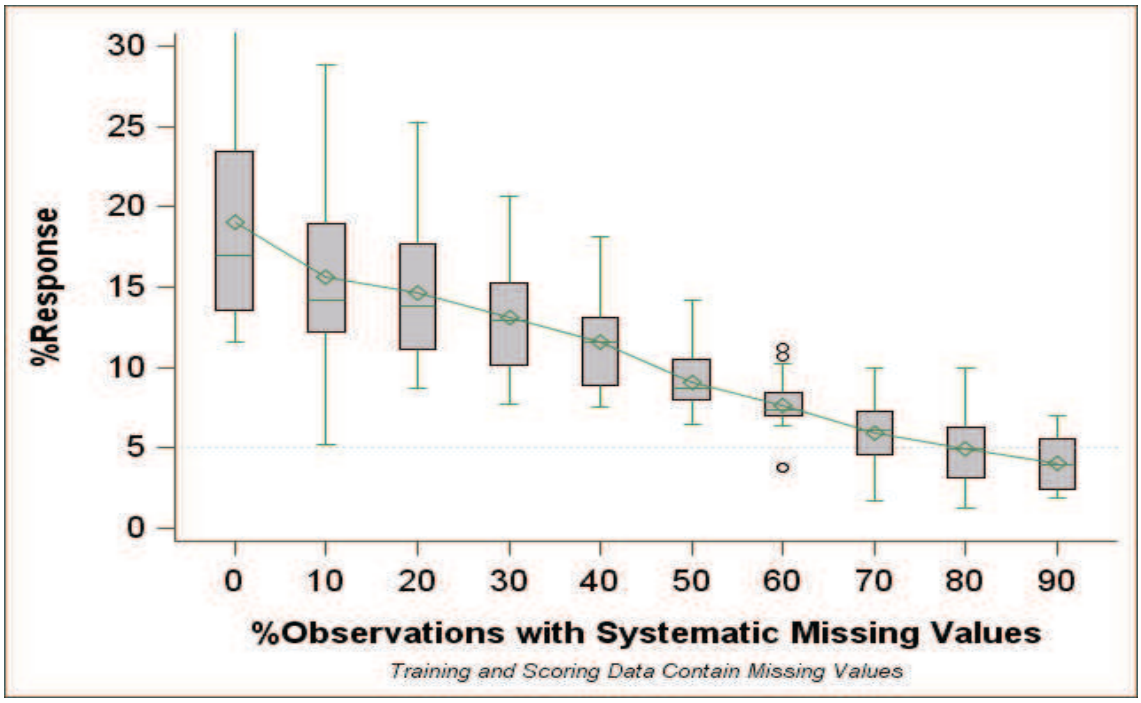

Abbildung 3: Simulationsergebnis für systematisch fehlende Werte

2 Grundlagen der Simulation in SAS

2.1 Allgemeines Simulations Template in SAS

Wicklin [2] präsentiert ein allgemeines Template für die Simulation univariater Daten in einem SAS Datastep. Dieses Template lässt sich einfach an die individuellen Anforderungen anpassen und erweitern.

```
%let N=100;
data Sample(keep=x);
 call streaminit(9876);
 do i = 1 to &N;
   x = rand("DistributionName",param1,param2, ...);
   output;
 end;
run;
```

Beachten Sie, dass hier, genauso wie von [3] Fan et.al vorschlagen, eine DO-Schleife in einem Dataset zum Einsatz kommt. Die entsprechenden Zufallszahlen und Verteilungen werden über die RAND-Funktion erzeugt.

2.2 Verwendung der „neuen“ Generatoren für Zufallszahlen

Die „alten“ Zufallszahlengeneratoren in SAS Base, wie zum Beispiel RANUNI, RANNOR, oder RANPOI verwenden einen älteren Algorithmus aus den 1970ern. Dieser Algorithmus ist auch bei den PROBxxx- oder xxxINV-Funktionen im Einsatz.

In diesen Funktionen wird ein Algorithmus verwendet, der eine kürzere Periode als der Mersenne-Twister Algorithmus hat. Der Vorteil einer sehr langen Periode, ist die lange Dauer (Sequenz) bis sich die Zufallszahlenreihenfolge wiederholt. Bei kleinen Stichproben im Bereich von Tausenden ist das kein Problem, bei Simulationen mit Millionen von Wiederholungen möglicherweise schon.

Dieser Algorithmus ist in der RAND Funktion seit der Version 9 in SAS implementiert. Der folgende Code zeigt ausgewählte Möglichkeiten des Aufrufs für unterschiedliche Verteilungen in einem SAS Datastep. Vor dem ersten Aufruf der RAND-Funktion muss das SEED mit CALL STREAMINIT initialisiert werden. Hinweis: positive Seed erzeugen eine Sequenz von Zufallszahlen, die immer wieder ident reproduziert werden kann. Bei nicht-positiven Seeds wird die Systemzeit des Computers zur Initialisierung der Zufallszahlenfolge verwendet.

```
call streaminit(9876);
 x1=rand("Bernoulli",0.5);          *** Münzwurf;
 x2=rand("Binomial",0.5,10);        *** Anzahl der Erfolge bei 10
                                    Versuchen;
 x3=rand("Geometric",0.5);          *** Anzahl der Versuche bis zum
                                    Erfolg;
```

```
 x4=ceil(6*rand("Uniform"));        *** Ergebnisse eines Würfels;
 x5=rand("Table",0.5,0.3,0.2);      *** Häufigkeiten mit Zurücklegen;
 x6=rand("Poisson",4);              *** Anzahl der Ereignisse pro
                                    Zeitintervall;
 x7=rand("Uniform");                *** Gleichverteilung im Interval
                                    [0,1];
 x8=rand("Normal",24,6);            *** Normalverteilung mit mu=24 und
                                    sigma=6;
```

Dieser Algorithmus steht auch in der SAS IML Software zur Verfügung. Dort ist in der Funktion RANDGEN implementiert, wie hier ausgewählte Beispiele zeigen.

```
call randseed(7654);
 call randgen(x_bern,"Bernoulli",0.5);
 call randgen(x_binom,"Bimonial",0,5,10);
 Table_prob={0.5 0.3 0.2};
 call randgen(x_table,"Table",table_prob);
```

2.3 Vermeidung von SAS-Macro Loops

Rein technisch gesehen können Sie SAS Macro Loops für Simulationen, wie im folgenden Code dargestellt, verwenden. Davon wird aber abgeraten, da in SAS mit DO Loops in einem SAS Datastep deutlich bessere Performance erreicht werden kann. Es mag zwar Beispiele geben, wo separate Simulationsschleifen in SAS Makros nötig sind. Generell wird von Implementierungen wie der folgenden aber abgeraten. Siehe auch [2]:

```
%macro Simulate(N, NumSamples);
proc datasets nolist;
delete OutStats; /* delete data if it exists */
run;

%do i = 1 %to &NumSamples;
   data Temp; /* create one sample */
    call streaminit(0);
    do i = 1 to &N;
      x = rand("Uniform");
      output;
    end;
  run;

proc means data=Temp noprint; /* compute one statistic */
 var x;
 output out=Out mean=SampleMean;
run;

proc append base=OutStats data=Out; /* accumulate statistics */
run;

%end;
%mend;
```

Eine bessere Version ist es, den Loop in einem SAS Datastep zu implementieren und die Analyse der Simulationsergebnisse in den folgenden Prozeduren mit einem BY-Statement durchzuführen.

```
%macro Simulate(N, NumSamples);

   data Temp;
   call streaminit(0);
   do SimulationRun = 1 to &NumSamples;
        do i = 1 to &N;
           x = rand("Uniform");
           output;
        end;
    end;
     run;

   proc means data=Temp nway noprint;
      class SimulationRun;
      var x;
      output out=Out mean=SampleMean;
   run;

%mend;
```

3 Mathematische Programmierung in SAS mit SAS IML Software

3.1 Die Mächtigkeit und Vorteil von SAS IML

SAS bietet mit dem SAS Datastep und der SAS Language mächtige Möglichkeiten der Datenanalyse und des Datenmanagements. Es gibt aber viele Fälle, wo der Zugriff auf Daten über eine Matrixsprache von Bedeutung ist. SAS bietet diese Möglichkeiten mit der SAS IML Software.

Diese ist voll in das SAS System integriert und erlaubt das Einlesen von SAS Datasets, das Ausgeben von Ergebnisobjekten nach SAS und die Verwendung von SAS Funktionen und SAS Formaten. SAS IML bietet Operationen u.a. für Matrizen, Matrixmultiplikationen, Vektoren, Skalare, Teilmatrizen, Indizes und vieles mehr. SAS IML bietet auch eine Integration zwischen SAS und dem R Open Source Project. Abbildung 4 gibt eine Übersicht über Operatoren in der SAS IML Sprache.

Operator	Description
` (accent grave)	Transpose (postfix)
- *(prefix)*	Negative prefix
[ ]	Subscript
**	Matrix exponentiation
##	Element-wise exponentiation
*	Matrix multiplication
#	Element-wise multiplication
/	Element-wise division
@	Direct (Kronecker) product
+	Addition
-	Subtraction
||	Horizontal concatenation
//	Vertical concatenation

Abbildung 4: Operatoren in der SAS IML Software

Damit können zum Beispiel folgende Operationen durchgeführt werden:

- A+B : matrix addition
- A*B : matrix multiplication,
- A#B : element-wise multiplication
- A[5,2] : Element aus der 5. Zeile, 2. Spalte
- A[1:3,2:10] :die ersten drei Spalten für die 2. bis 10. Zeile
- W = INV(T(x)*x): Inversion und Transposition von Matrizen und Vektoren

Wicklin [4] gibt einen detaillierten Einblick in die Möglichkeiten der SAS IML Language.

3.2 Simulation von Daten einer Multivariaten Verteilung

Simulationen in SAS IML können beispielsweise auf folgende Art durchgeführt werden. Dieses Programm ist in [5] Wicklin dargestellt und wurde adaptiert.

```
proc iml;
 Mean = {42, 5200, 280}; /* population means */
 Cov =
```

```
 {12 48 25, /* population covariances */
 48 420 0,
 25 0 100};
 N = 1000; /* sample size */

call randseed(123);
X = RandNormal(N, Mean, Cov); /* x is a 1000 x 3 matrix */
SampleMean = mean(X);
SampleCov = cov(X);
varNames = {Alter Volumen Events};

print SampleMean[colname=varNames],
SampleCov[colname=varNames rowname=VarNames];

/* write sample to SAS data set for plotting */
create MVN from X[colname=varNames]; append from X; close MVN;
quit;
```

Dabei werden Zufallszahlen einer multivariaten Normalverteilung simuliert. Der Vektor für die Mittelwerte und die Varianz-Kovarianz-Matrix werden direkt im SAS IML als solche angegeben bzw. könnten auch aus SAS Datasets gelesen werden. Die IML Funktion RANDNORMAL verwendet diese Objekte, um eine Matrix von Simulationsdaten zu erzeugen.

Die Ergebnisse können in SAS IML oder mit SAS Prozeduren ausgewertet werden, wie hier zum Beispiel mit PROC CORR.

```
/* create scatter plot matrix of simulated data */
proc corr data=MVN plots(maxpoints=NONE)=matrix(histogram);
run;
```

Abbildungen 5 und 6 zeigen die entsprechenden Ergebnisse.

Simple Statistics

Variable	N	Mean	Std Dev	Sum	Minimum	Maximum
ALTER	1000	41.96466	3.50861	41965	30.27959	53.47730
VOLUMEN	1000	5200	20.59750	5199748	5129	5266
EVENTS	1000	280.08244	10.29895	280082	248.16537	314.96263

Pearson Correlation Coefficients, N = 1000
Prob > |r| under H0: Rho=0

	ALTER	VOLUMEN	EVENTS
ALTER	1.00000	0.66484 <.0001	0.72743 <.0001
VOLUMEN	0.66484 <.0001	1.00000	-0.00653 0.8365
EVENTS	0.72743 <.0001	-0.00653 0.8365	1.00000

Abbildung 5: Output der CORR Procedure

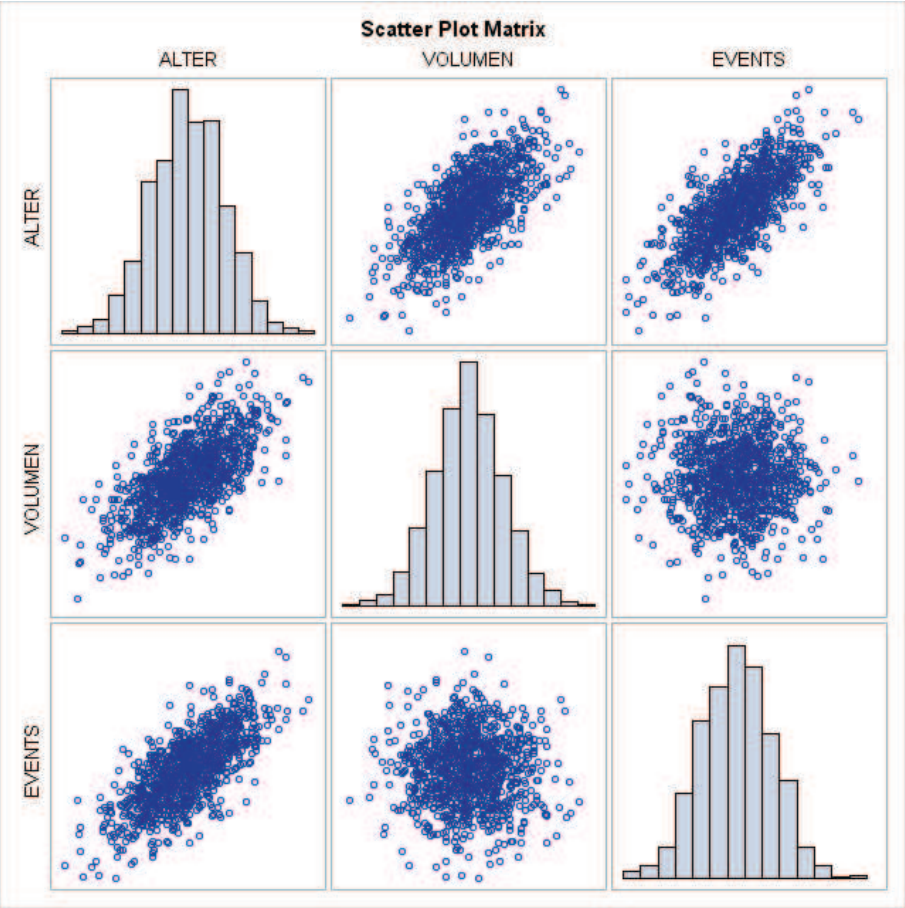

Abbildung 6: Scatterplot-Graphik der CORR Procedure

3.3 Simulationen mit SAS IML

Wir haben gesehen, dass die RAND Funktion im Datastep sehr mächtig für die Simulation von Daten für univariate Verteilungen ist. SAS IML hingegen ist das Werkzeug der Wahl für die Simulation von korrelierten Daten und von multivariaten Verteilungen.

SAS IML beinhält viele built-in Funktionen für die Simulation unterschiedlicher univariater und multivariater Verteilungen. SAS IML unterstützt auch jene Matrix-Berechnungen, um Datenstichproben von weniger häufig verwendeten Verteilungen zu ziehen.

Hinweis: für den oben gezeigten Spezialfall der multivariaten Normalverteilung bietet auch SAS STAT mit der SIMNORMAL Prozedur eine Simulationsmöglichkeit.

4 Weitere Simulationsbeispiele

4.1 Simulation von Daten aus einer Kombination von Verteilungen

Folgendes Beispiel zeigt die Simulationen von Daten, die aus unterschiedlichen Verteilungen kombiniert werden.

In einem Call Center werden die Anrufe in 3 Gruppen geteilt:

- 50% sind einfache Anfragen
- 20% sind spezialisierte Anfragen und
- 30 % sind „harte Fälle“.

Die Erfahrungswerte bzgl. der Bearbeitungsdauer sind in Tabelle 1 dargestellt.

Tabelle 1: Bearbeitungsdauer nach Anfragetyp

Question	Mean	Standard Deviation
Easy	3	1
Specialized	8	2
Hard	10	3

Folgendes SAS Datastep Programm zeigt, wie diese Daten aus einer Verteilung mit 3 Kategorien und einer Normalverteilung simuliert werden können.

```
data Calls(drop=i);
call streaminit(0);
array prob [3] _temporary_ (0.5 0.2 0.3); /* mixing probabilities */

do i = 1 to 1000;
     Type = rand("Table", of prob[*]); /* returns 1, 2, or 3 */
     if      Type=1 then time = rand("Normal", 3, 1);
     else if Type=2 then time = rand("Normal", 8, 2);
     else                time = rand("Normal", 10, 3);
     output;
end;
run;

proc univariate data=Calls;
ods select Histogram;
histogram time / vscale=proportion kernel(lower=0 c=SJPI);
run;
```

Abbildung 7 zeigt das Histogramm mit Kern Schätzer, das mit PROC UNIVARIATE erstellt wurde.

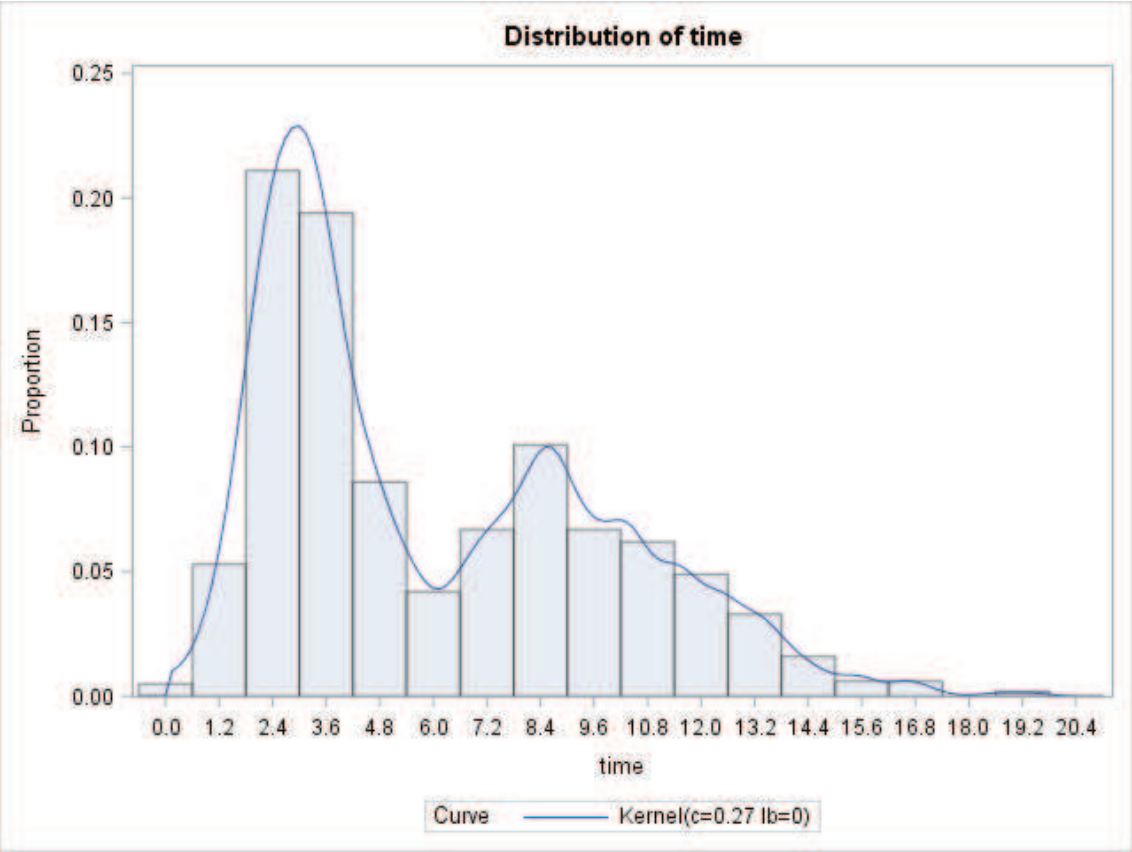

Abbildung 7: Histogramm mit Kern Schätzer

4.2 Simulation von Daten aus komplexen Verteilungen

Wicklin [5] beschreibt, wie Daten aus komplexen Verteilungen simuliert werden können, auch wenn diese im Basis-Set der 20 Verteilungen für RAND Funktion nicht enthalten sind:.

- das Löschen von bestimmten Wertebereichen einer Verteilung ergibt eine Truncated Distribution,
- Transformationen können verwendet werden, um eine Verteilung in eine andere zu transformieren, zum Beispiel die Normalverteilung in eine Log-Normalverteilung
- innerhalb der gleichen Verteilungsfamilie können Verschiebung und Skalierung verwendet werden, um die gewünschten Verteilungen zu erhalten.

5 Optimierung von Simulationen

5.1 Beschleunigung der Simulationsläufe durch Unterdrückung von Output

Bei Simulationen ist man typischerweise an den erzeugten Daten im SAS Dataset und weniger an den Ergebnissen im Output-Fenster oder den Graphiken interessiert. Daher können Optionen wie NOPRINT oder PLOTS=NONE hier hilfreich sein, um die Outputmenge einzuschränken und die Laufzeit zu verringern.

Weiters sind folgende Schritte ideal um den Output abzuschalten und so die Laufzeit zu verbessern:

- Erstellung der Graphiken abschalten: ODS GRAPHICS OFF;
- Über ODS alle Ergebnisse unterdrücken: ODS EXCLUDE ALL;
- Den Tree-View im Results-Fenster nicht befüllen: ODS RESULTS OFF;
- Die Notes im Log unterdrücken: OPTIONS NONOTES;

Wicklin [5] präsentiert dazu die folgenden beiden Makros:

```
%macro ODSOff(); /* call prior to BY-group processing */
ods graphics off;
ods exclude all;
ods results off;
options nonotes;
%mend;

%macro ODSOn(); /* call after BY-group processing */
ods graphics on;
ods exclude none;
ods results on;
options notes;
%mend;
```

5.2 Planung der Simulations-Studie vor dem Start

Bevor Sie Ihre Simulationsstudie starten, sollten Sie neben der inhaltlichen Planung auch operative Schritte planen:

- Starten Sie die ersten Testläufe zur inhaltlichen Programmverifikation mit nur 2-5 Iterationen. Hier geht es um die Korrektheit und die Validierung ihres Programms. Wichtig: sie sollten hier mehr als einen Durchlauf durchführen.
- Starten Sei den ersten Performancetest mit 100 – 1000 Iterationen. So bekommen Sie einen Eindruck über die Laufzeit Ihrer Simulationen. So können Sie abschätzen, welche Laufzeit und welcher Speicherplatzbedarf bei einem Durchlauf mit 100.000 oder Millionen von Iterationen zu erwarten ist.
- Stellen Sie sicher, dass sie ihr Programm vor dem „Submit“ speichern!!! So können sie notfalls die Session ohne Verluste vollständig abbrechen.
- Starten Sie mit einem „groben“ Grid und verfeinern Sie dort, wo Sie mehr Details benötigen. Im zweidimensionalen Fall benötigt ein 20x20 die vierfache Laufzeit eines 10x10 Grids.

Literatur

[1] G. Svolba: Data Quality for Analytics Using SAS: SAS Press 2012, Cary NC

[2] R. Wicklin: Simulating Data With SAS: SAS Press 2013, Cary NC

[3] X. Fan, A. Felsövalyi, S. Sivo, S. Keenan: SAS for Monte Carlo Studies: SAS Press 2002, Cary NC

[4] R. Wicklin: Statistical Programming with SAS/IML Software: SAS Press 2010, Cary NC

[5] R. Wicklin: Ten Tips for Simulating Data with SAS : Paper SAS1387-2015, Proceedings des SAS Global Forums 2015.